RECEIVED
CENTRAL FAX CENTER

AUG 0 2 2006

AMENDMENTS TO THE CLAIMS

This listing of claims will replace all prior versions and listings of claims in the application.

Listing of Claims:

Claim 1 (Currently Amended): A processor-based method performed by software emulating an instruction processor, the method comprising:

processing read instructions with an emulated processor executing within an emulation environment to output independent read requests via an operand interface and an op-code interface of the emulated processor to independently fetch op-codes and operands from an emulated memory external from the emulated processor;

independently comparing op-code reference data and operand reference data to operands and op-codes received in response to the read requests; and

recording results of the independent comparisons.

Claim 2 (Original): The method of claim 1, wherein independently comparing further comprises:

storing the op-code reference data and the operand reference data within a set of data memories of the emulated instruction processor;

maintaining within the emulated instruction processor an operand data pointer to address the operand reference data and an op-code pointer to address the op-code reference data; and

independently accessing the operand reference data with the operand data pointer and the op-code reference data with the op-code data pointer during processing of the read instructions to verify the received op-codes and the received operands.

Claim 3 (Original): The method of claim 2, further comprising:

when processing the read instructions, determining whether each of the read instructions required an operand read request or an op-code read request; and

independently updating the op-code reference pointer or the operand reference pointer based on the determination.

Claim 4 (Original): The method of claim 2, wherein the read instructions form part of an instruction stream executed by the emulated instruction processor, the method further comprising:

processing a flow control instruction of the instruction stream with the emulated instruction processor; and

upon processing the flow control instruction, synchronizing the op-code reference pointer and the operand reference pointer to respectively address a portion of the op-code reference data and a portion of the operand reference data associated with a target address of the flow control instruction.

Claim 5 (Original): The method of claim 2, compiling test software to output the operand reference data, the op-code reference data, and the instruction stream.

Claim 6 (Original): The method of claim 2, wherein independently comparing further comprises:

latching the op-code reference data within a first latch within the emulated instruction processor; and

comparing the latched op-code referenced data and the received op-codes with a comparator to produce the results.

Claim 7 (Original): The method of claim 2, wherein independently comparing further comprises:

latching the operand reference data within a first latch within the emulated instruction processor; and

comparing the latched operand referenced data and the received operand with a comparator to produce the results.

Claim 8 (Original): The method of claim 1, further comprising:

storing write data within a data memory of the emulated instruction processor; maintaining within the emulated instruction processor a write data pointer to address the write data; and

processing a write instruction with the emulated processor to output a write request via a data interface of the emulated processor, wherein the write request comprises a portion of the write data referenced by the write pointer.

Claim 9 (Original): The method of claim 1, further comprising generating a report based on the results, wherein the report identifies any of the received op-codes that do not match the op-code reference data and any of the received operands that do not match the operand reference data.

Claim 10 (Original): The method of claim 1, wherein recording results comprises storing addresses associated with received operands and the op-codes, copies of the received operands or op-codes, copies of the reference operands and the reference op-codes, or copies of the instruction.

Claim 11 (Original): The method of claim 1, wherein recording the results comprises recording the results within a register within the emulated instruction processor.

Claim 12 (Original): The method of claim 1, further comprising applying bit masks to at least a portion of a result of the independent comparisons of the op-code reference data and the operand reference data to the op-code and the operand received in response to the read requests.

Claim 13 (Original): The method of claim 1, further comprising:

storing the addresses of a small number of received operands and received op-codes within a cache within the emulated instruction processor; and

selectively enabling and disabling access to the cache when executing subsequent read instructions and waiting to output read requests via the operand interface and the op-code interface when the read instructions request operands and op-codes are invalidated within the cache based on a configurable option.

Claim 14 (Currently Amended): A processor-based system for emulating an instruction processor comprising:

a computing system to provide an emulation environment; and

software executing within the emulation environment to emulate an instruction processor having an operand interface and an op-code interface and to emulate a memory external to the instruction processor,

wherein the software emulates the instruction processor by processing read instructions and outputting corresponding read requests on the operand interface or the op-code interface to independently fetch respective op-codes or operands from the emulated memory, and independently comparing op-code reference data and operand reference data to operands and op-codes received from the operand interface and op-code interface in response to the read requests.

Claim 15 (Original): The system of claim 14, wherein the software emulates the instruction processor by recording results of the independent comparisons within a register of the emulated instruction processor.

Claim 16 (Original): The system of claim 14, wherein the emulated instruction processor comprises:

a first data memory to store the op-code reference data; and

a second data memory to store the operand reference data.

Claim 17 (Original): The system of claim 16, wherein the emulated instruction processor comprises:

a control unit that maintains an operand data pointer to address the operand reference data within the first data memory and an op-code pointer to address the op-code reference data within the second data memory,

wherein the control unit independently accesses the operand reference data with the operand data pointer and the op-code reference data with the op-code data pointer during processing of the read instructions to verify the received op-codes and the received operands.

Claim 18 (Original): The system of claim 17, wherein upon processing the read instructions, the control unit determines whether each of the read instructions required an operand read request or an op-code read request, and independently updates the op-code reference pointer or the operand reference pointer based on the determination.

Claim 19 (Original): The system of claim 17, wherein the read instructions form part of an instruction stream executed by the emulated instruction processor, and upon processing a flow control instruction of the instruction stream, the control unit synchronizes the op-code reference pointer and the operand reference pointer to respectively address a portion of the op-code reference data and a portion of the operand reference data associated with a target address of the flow control instruction.

Claim 20 (Original): The system of claim 19, further comprising a compiler executing on the computing system to compile test software to output the operand reference data, the op-code reference data, and the instruction stream for execution by the emulated instruction processor.

Claim 21 (Original): The system of claim 19, wherein the emulated instruction processor further comprises:

a latch to latch the op-code reference data from the first data memory; and

a comparator to compare the latched op-code referenced data and the received op-codes.

Claim 22 (Original): The system of claim 19, wherein the emulated instruction processor further comprises:

a latch to latch the operand reference data from the second data memory; and

a comparator to compare the latched operand referenced data and the received operands.

Claim 23 (Original): The system of claim 14, wherein the emulated instruction processor further comprises:

a data memory to store write data; and

a control unit to maintain a write data pointer to address the write data, wherein the control unit processes a write instruction to output a write request via a data interface of the emulated processor, and further wherein the control unit generates the write request to comprises a portion of the write data referenced by the write pointer.

Claim 24 (Original): The system of claim 14, further comprising emulation control software executing on the computing system to generate a report that presents/identifies any of the received op-codes that do not match the op-code reference data and any of the received operands that do not match the operand reference data.

Claim 25 (Original): The system of claim 14, wherein the emulated instruction processor further comprises:

a set of memories to store bit masks; and

bit masks to compare results of the comparison of the received operands and the received op-codes to the reference operands and the reference op-codes to mask portions of these comparisons.

Claim 26 (Original): The system of claim 14, wherein the emulated instruction processor further comprises a cache to store the addresses of the received operands and received op-codes, wherein the emulated instruction processor selectively waits to issue read requests for subsequent read instructions via the operand interface and the op-code interface until the read instructions request operands and op-codes become invalidated within this cache based on a configurable option.

Claim 27 (Currently Amended): A processor-based system for emulating an instruction processor comprising:

compiling means for compiling test software to produce operand reference data, op-code reference data, and an instruction stream having read instructions; and

emulating means for emulating an instruction processor having an operand interface and an op-code interface and for emulating a memory external to the instruction processor, wherein the emulating means comprises:

controlling means for controlling the emulated instruction processor to process the read instructions and output corresponding read requests on the operand interface or the op-code interface to independently fetch respective operands or op-codes from the emulated memory,

receiving means for receiving operands and op-codes from the operand interface and op-code interface in response to the read requests, and

comparing means for independently comparing the op-code reference data and the operand reference data to the received.

Claim 28 (Original): The system of claim 27, wherein the emulating means further comprises:

a first storing means for storing the op-code reference data; and

a second storing means for storing the operand reference data.

Claim 29 (Original): The system of claim 27, wherein the controlling means comprises:

first referencing means for addressing the operand reference data within the first storing means; and

second referencing means for addressing the op-code reference data within the second storing means.

Claim 30 (Original): The system of claim 27, further comprising reporting means for generating a report that presents and identifies any of the received op-codes that do not match the op-code reference data and any of the received operands that do not match the operand reference data.